Deploy a React application on AWS using Amplify, Cognito, GitHub and setup a CI/CD pipeline

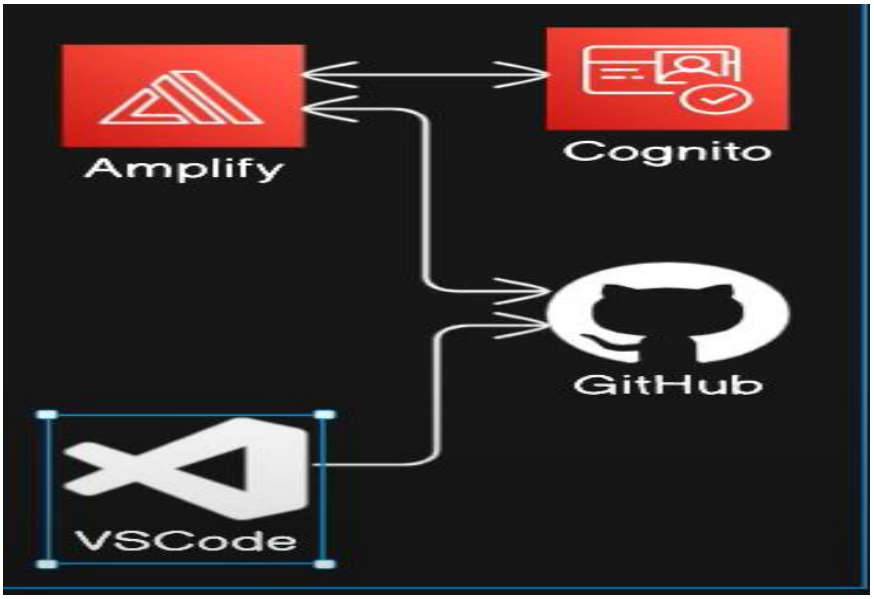
Objective: The goal of your project is to leverage AWS services, including Amplify, Cognito, and GitHub CI/CD, to create a React application with seamless deployment, robust authentication, and efficient continuous integration and deployment processes. This approach aims to streamline development workflows, enhance security, and provide a scalable infrastructure for your application.

- **AWS Amplify**: AWS Amplify simplifies app development by providing tools for hosting, authentication, data storage, APIs, and more. It integrates with Git for CI/CD, supports both static and dynamic content, and offers real-time data and offline capabilities. It streamlines deployment, enhances security, and scales applications efficiently.
- **AWS Cognito:** AWS Cognito manages user authentication and authorization for web and mobile apps. It supports user sign-up, sign-in, and multi-factor authentication, and integrates with social identity providers. It also handles user management, including password recovery and account verification, ensuring secure and scalable user access control.

- **IAM:** AWS Identity and Access Management (IAM) manages permissions and access to AWS services and resources. It allows you to create users, groups, and roles, set policies to control permissions, and enforce security best practices. IAM ensures secure and controlled access to AWS, protecting your resources and data.
- **GIT:** Git is a version control system that tracks changes to code, enabling collaboration among developers. It allows branching, merging, and reverting changes, providing a history of modifications. Git helps manage code versions, facilitates collaborative development, and ensures consistent code integration and deployment in projects.

This combination of tools allows you to focus on building your React application's core features while ensuring a secure and efficient development workflow.

So, this is the architecture we are going to build:



Step 1: Setting up the environment

- I'm using the Visual Studio Code as IDE you can use other applications as IDE.
- Download Visual studio from the link (https://nodejs.org/en/download)
- Open the Visual Studio Code as IDE.

Now first of all we need to setup the Amplify CLI, we will use the npm package to do it, so open a terminal and run this:

```
npm install -g @aws-amplify/cli
```

Next we need to configure Amplify to make the CLI work with your AWS account, we will have to setup a new user, configure access keys and so on. So run:

```
amplify configure
```

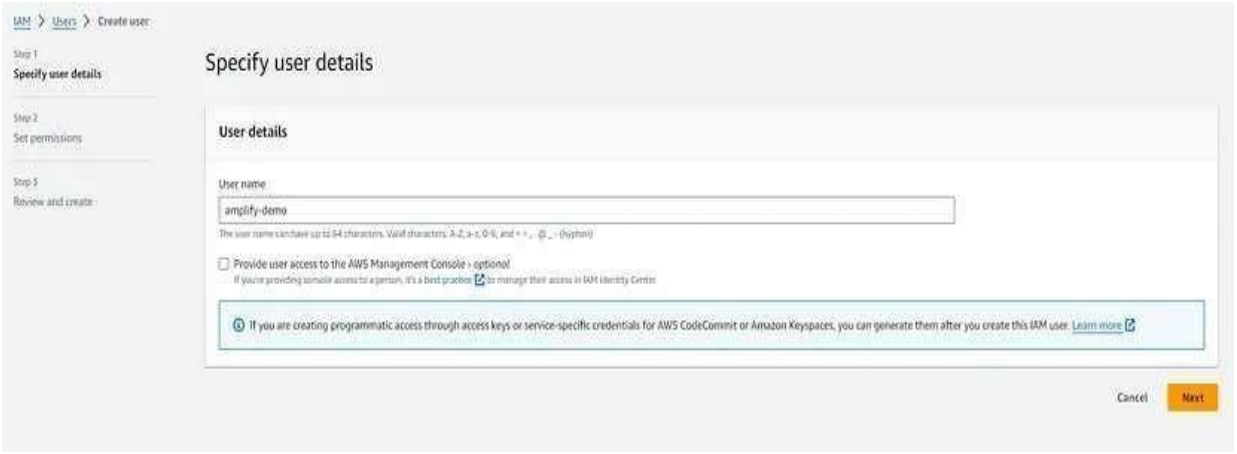
It will ask you to sign in as an AWS administrator account, so type in the credentials for a user that has admin permissions but not your root account. If you haven't already created this user, you can follow along this guide:

After you have signed in, select your region by moving up and down with your arrow keys. Choose the one that is closest to you to have minimal latency, I will choose "us-west-1" which corresponds to "Virginia".

Click enter and this will show up:

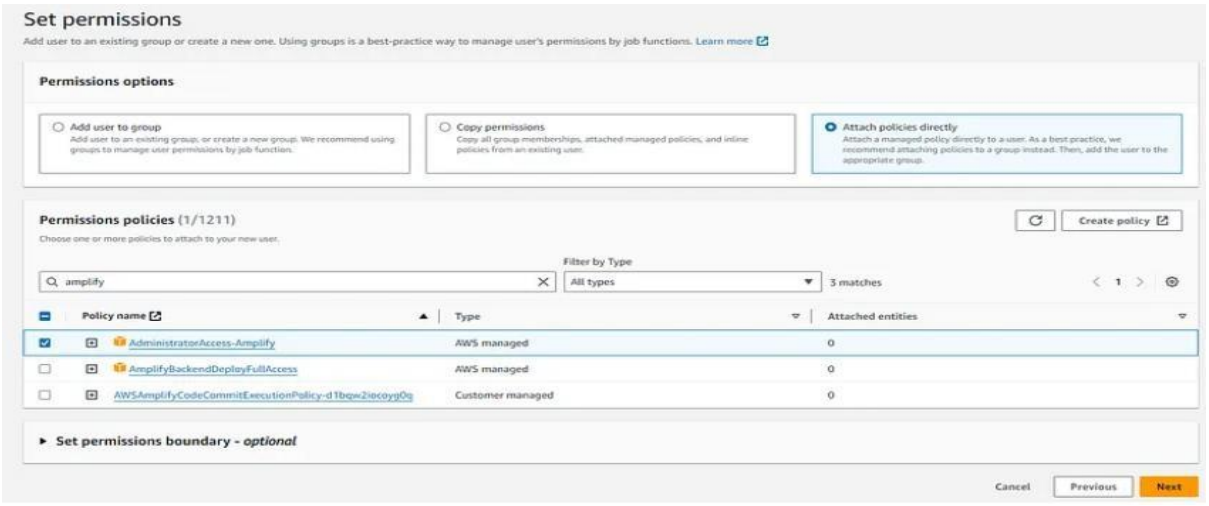
```
Specify the AWS Region
? region:  eu-west-3
Follow the instructions at
https://docs.amplify.aws/cli/start/install/#configure-the-amplify-cli

to complete the user creation in the AWS console
https://console.aws.amazon.com/iamv2/home#/users/create
Press Enter to continue
```

In the meantime this page will open automatically in your browser to create a new user:



If not, navigate to the “IAM” service and select “Users” > “Create a user”, now it should open the same window.

Create a new user, call it whatever you want and click “Next”. Then choose the option “Attach policies directly” and type “amplify” in the search bar and select “AdministratorAccess-Amplify”:

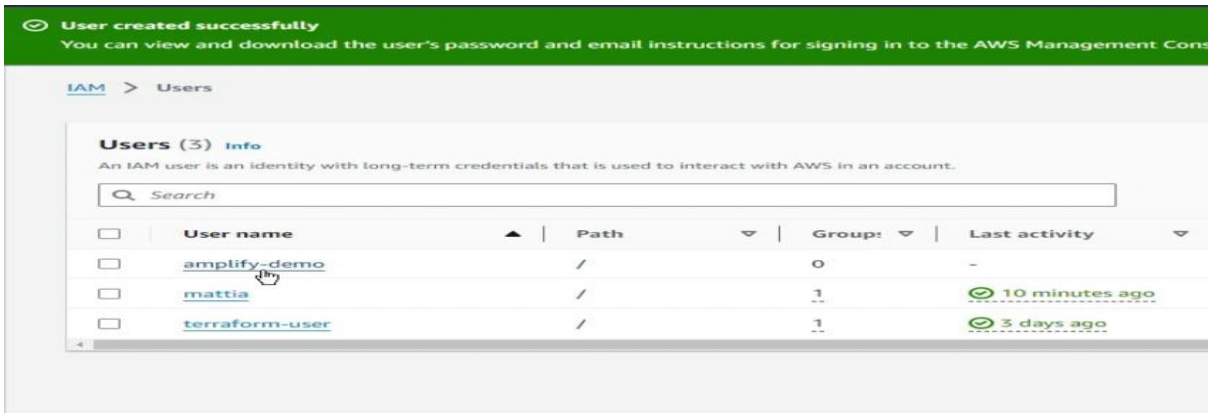


Then click “Next” > “Create user”.

Back in the terminal, if you press Enter it will ask you for your access key:

```
Specify the AWS Region
? region:  eu-west-3
Follow the instructions at
https://docs.amplify.aws/cli/start/install/#configure-the-amplify-cli

to complete the user creation in the AWS console
https://console.aws.amazon.com/iamv2/home#/users/create
Press Enter to continue

Enter the access key of the newly created user:
? accessKeyId:  [hidden]
```

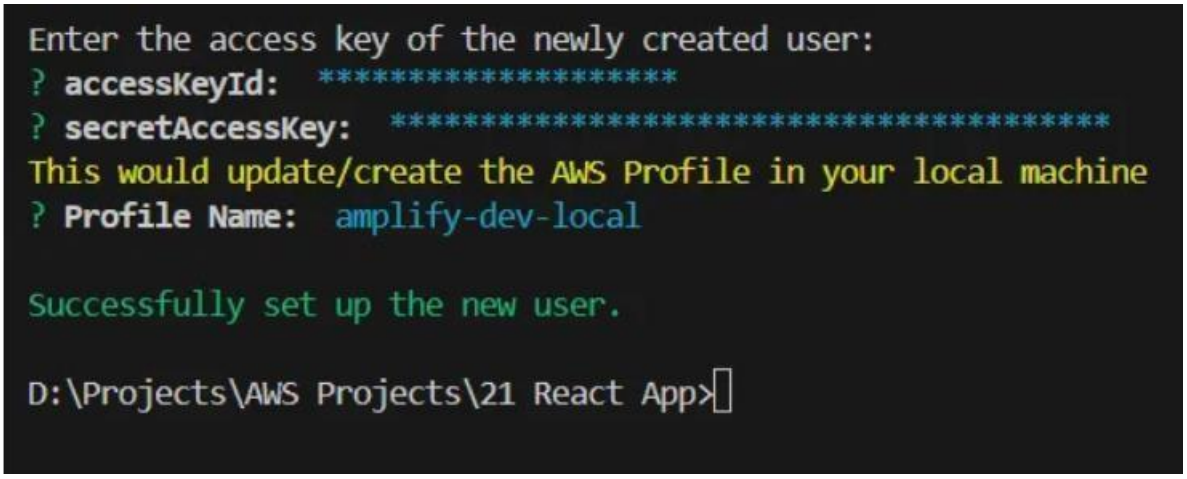
So back in the AWS console in the IAM dashboard select the newly created user:



Then go to the “Security credentials” tab and under “Access keys” click on “Create access key”.

As Use case select the first option (“Command Line Interface (CLI)”), tick the confirmation checkbox “I understand…” and click on Next and then create the access key. Copy the access key and paste it in the terminal where it asked for your access key. Then press Enter and do the same for the secret access key.

Now it will ask you to create the AWS Profile to save all the information we just put (the region, the access key, and so on) on your local PC, so choose a name:



Step 2: create the React application

Now we are ready to create our application. In the terminal run the following command:

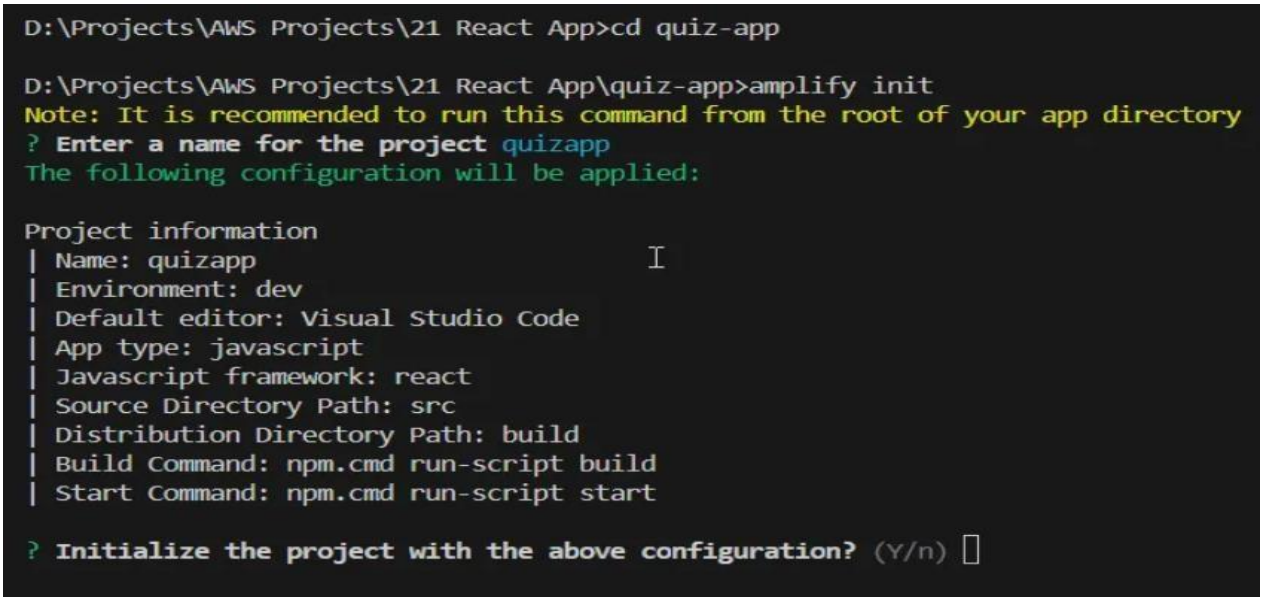
```
npx create-react-app quiz-app
```

This will create a vanilla React application. Now we will move to the folder of the application and run “amplify init” to initialize the Amplify project, it will ask some information to create our project

like the name, the programming language and so on, but some information will ne automatically detected. So to do this run:

```
cd quiz-app
amplify init
```

This will be displayed:

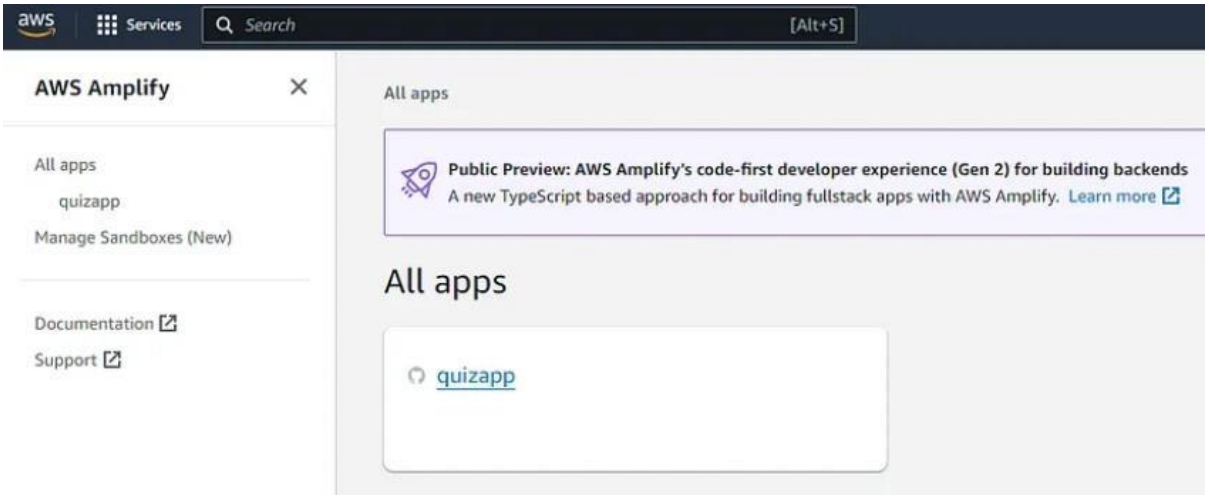


Now type “Y” to initialize the project with the above configuration and press Enter.

It will ask you the authentication method to use, you have to select “AWS profile” and then “amplify-dev-local” (the profile we created before, if you chose another name then select it).

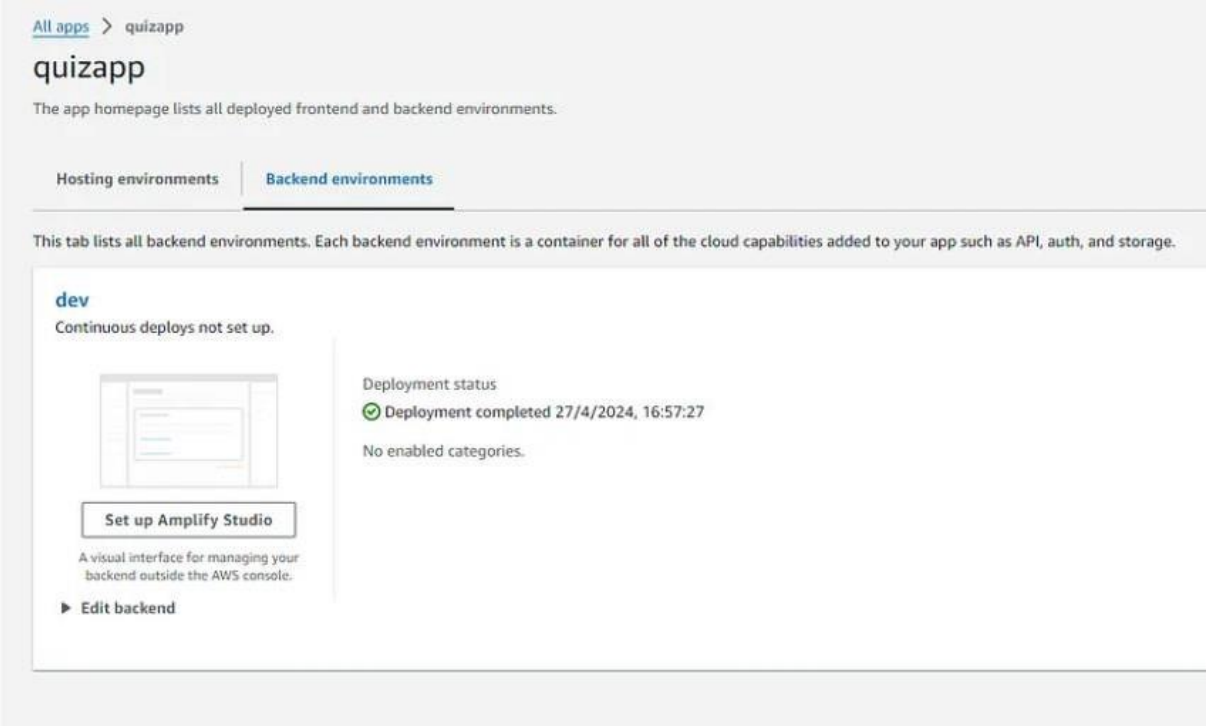
Next it will show a message about helping improve Amplify CLI by sharing configurations, type “N”.

Then check in the AWS console if the project has been set up successfully, go to the AWS service “Amplify” and you should see your application:



If you don’t see it, please make sure you are in the right AWS region, the one that you selected before (change it on the top-right on the AWS console).

If you click on the application it will open the overview, under the tab “Backend environments” there is an environment called “dev” that AWS created for us, but it is basically empty:



Step 3: Add authentication with Cognito

The next step is to add authentication to the React app using Cognito, another AWS service. To add Cognito, in the terminal simply run:

```
amplify add auth
```

It will ask if we want to use the default configuration, select “Default configuration” as we do not want to set up Google or Facebook authentication or to implement it manually.

Then we want users to sign in using their email, so select “Email”.

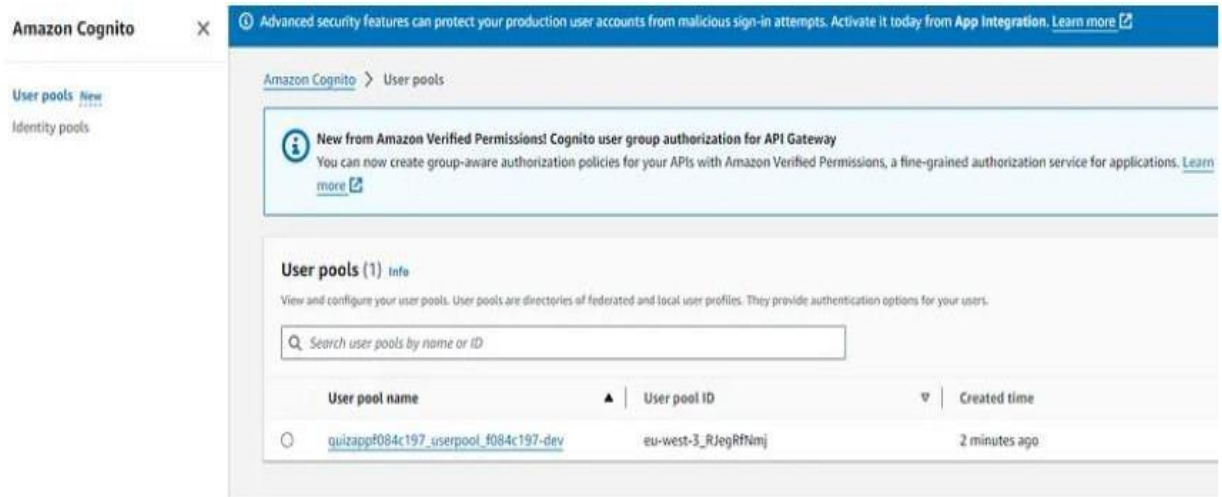
Next it will ask if you want to configure advanced settings, select “No, I am done.”

To push these changes to AWS run:

```
amplify push
```

It will ask you for confirmation, type “Y” to confirm your push action.

Now if you go to the AWS service “Amazon Cognito” in the AWS management console you should see a new user pool created for us:



If you click on it you can see that we do not have any users yet. So let's set up the UI so that we can create an account and log in as a user. So go to the "src/App.js" file in your React application and paste this code:

```
import React from 'react';

import './App.css';


// Imports the Amplify library from 'aws-amplify' package. This is used to configure your app to interact with AWS services.

import {Amplify} from 'aws-amplify';


// Imports the Authenticator and withAuthenticator components from '@aws-amplify/ui-react'.

// Authenticator is a React component that provides a ready-to-use sign-in and sign-up UI.

// withAuthenticator is a higher-order component that wraps your app component to enforce authentication.

import { Authenticator, withAuthenticator } from '@aws-amplify/ui-react';


// Imports the default styles for the Amplify UI components. This line ensures that the authenticator looks nice out of the box.

import  '@aws-amplify/ui-react/styles.css';


// Imports the awsExports configuration file generated by the Amplify CLI. This file contains the AWS service configurations (like Cognito, AppSync, etc.) specific to your project.

import awsExports from './aws-exports';


// Imports the Quiz component from Quiz.js for use in this file.

import Quiz from './Quiz';
```

```
// Configures the Amplify library with the settings from aws-exports.js, which includes all the AWS service configurations for this project.
Amplify.configure(awsExports);


function App() {
  return (
    <div className="App">
      <Authenticator>
        {({ signOut }) => (
          <main>
            <header className='App-header'>
              {/* Quiz Component */}
              <Quiz />
              {/* Sign Out Button */}
              <button
                onClick={signOut}
                style={{
                  margin: '20px',
                  fontSize: '0.8rem',
                  padding: '5px 10px',
                  marginTop: '20px'
                }}
              >
                Sign Out
              </button>
            </header>
          </main>
        )}
      </Authenticator>
    </div>
  );
```

```
}

export default withAuthenticator(App);
```

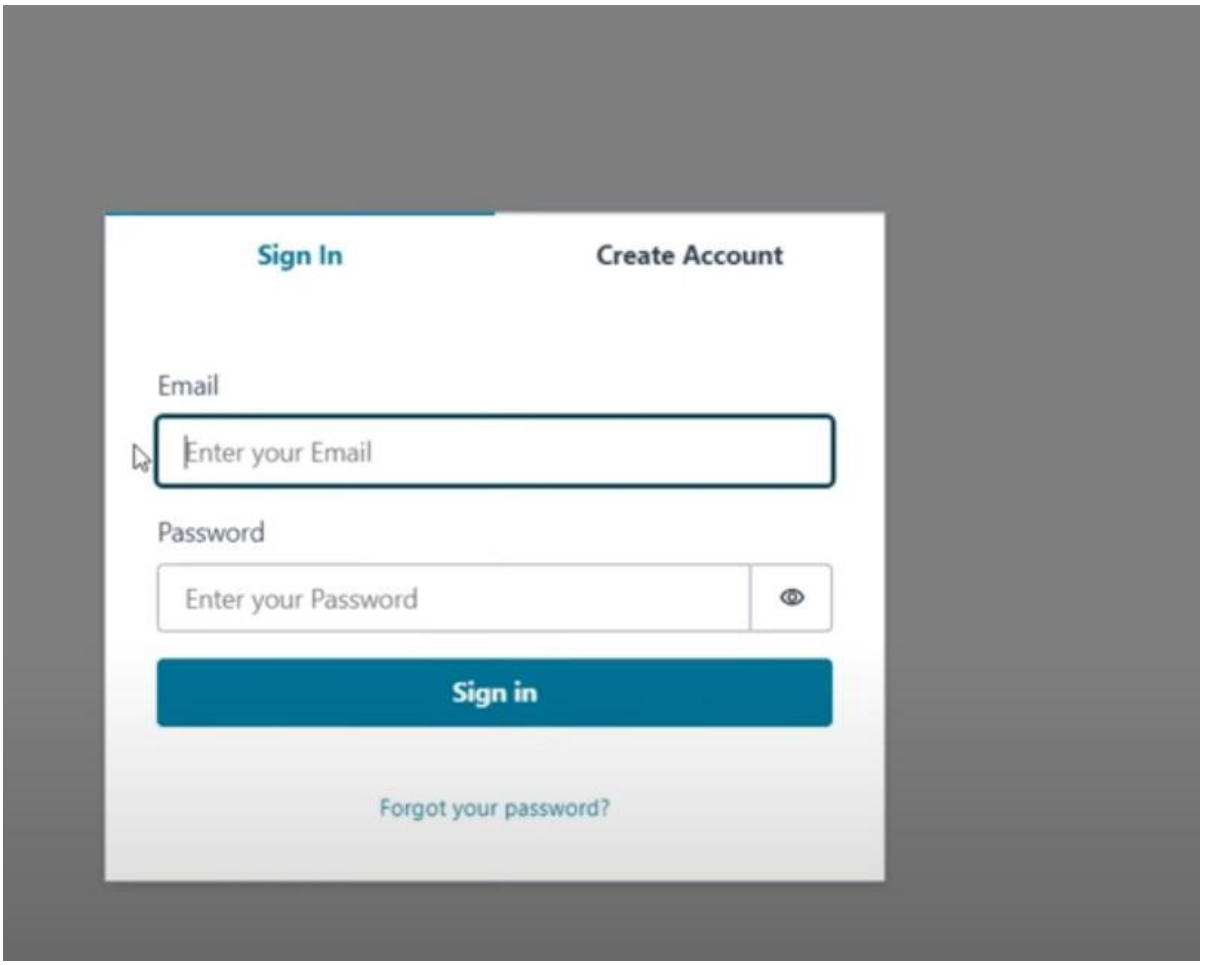
Then we need to install the Amplify libraries that give us the login user interface, so run the following command:

```
npm install aws-amplify @aws-amplify/ui-react
```

Then run this command to start your application:

```
npm start
```

A new tab in your browser will open, showing this login page:



Now click on “Create account”. Enter a valid email and a password, if you click on Next an authentication code will be sent to your email to activate your account, so enter the confirmation code and confirm.

You should now see this page as soon as you log in, with only a Sign Out button in the centre:

In the AWS console, if you refresh the Users section in the Amazon Cognito’s user pool, you should see now one user:

Step 4: add functionality and styling for quiz

Create a “Quiz.js” file in the “src” directory of your React application and paste this code:

```
import React, { useState } from 'react';
import quizData from './quizData';

function Quiz() {
  const [currentQuestion, setCurrentQuestion] = useState(0);
  const [score, setScore] = useState(0);
  const [showScore, setShowScore] = useState(false);
  const [selectedAnswer, setSelectedAnswer] = useState("");
  const [isCorrect, setIsCorrect] = useState(null);

  const handleAnswerOptionClick = (option) => {
    const correctAnswer = quizData[currentQuestion].answer;
    setSelectedAnswer(option);
    if (option === correctAnswer) {
      setScore(score + 1);
      setIsCorrect(true);
    } else {
      setIsCorrect(false);
    }

    // Delay moving to the next question to allow the user to see feedback
    setTimeout(() => {
      const nextQuestion = currentQuestion + 1;
      if (nextQuestion < quizData.length) {
```

```
      setCurrentQuestion(nextQuestion);
      setIsCorrect(null); // Reset for the next question
      setSelectedAnswer(""); // Reset selected answer
    } else {
      setShowScore(true);
    }
  }, 1000); // Adjust time as needed
};

return (
  <div className='quiz'>
    {showScore ? (
      <div className='score-section'>
        You scored {score} out of {quizData.length}
      </div>
    ) : (
      <>
        <div  className='question-section'>
          <div  className='question-count'>
            <span>Question {currentQuestion + 1}</span>/{quizData.length}
          </div>
          <div  className='question-text'>{quizData[currentQuestion].question}</div>
        </div>
        <div  className='answer-section'>
          {quizData[currentQuestion].options.map((option) => (
            <button
              onClick={() => handleAnswerOptionClick(option)}
              key={option}
              style={{ backgroundColor: selectedAnswer === option ? (isCorrect ? 'lightgreen' : 'pink') : '' }}
            >
              {option}
```

```
            </button>
          ))}
        </div>
        {selectedAnswer && (
          <div style={{ marginTop: '10px' }}>
            {isCorrect ? 'Correct! 🎉' : 'Sorry, that's not right. 😳'}
          </div>
        )}
      </>
    )}
  </div>
 );
}

export default Quiz;
```

We are fetching quiz questions and answers from a service called Open Trivia DB which exposes an API. You can customize your quiz as you want by changing the API options in https://opentdb.com/api_config.php and then based on the configuration it will generate the corresponding URL to replace in the code in "Quiz.js", for example for the one already provided in the code:

https://opentdb.com/api.php?amount=15&category=11&difficulty=hard&type=multiple

In this case in the API configuration I selected:

- **Number of questions:** 15
- **Category:** Entertainment Film
- **Difficulty:** Hard

- **Type:** Multiple Choice

Once you have finished with “Quiz.js”, in “App.js” you should uncomment these two lines:

- “import Quiz from ‘./Quiz’”
- <Quiz />

In addition to this, the application uses Tailwind CSS for the UI (https://tailwindcss.com/) and the “he” library, which is a robust HTML entity encoder/decoder to resolve ambiguous sequences of characters returned by the API in the questions and answers.

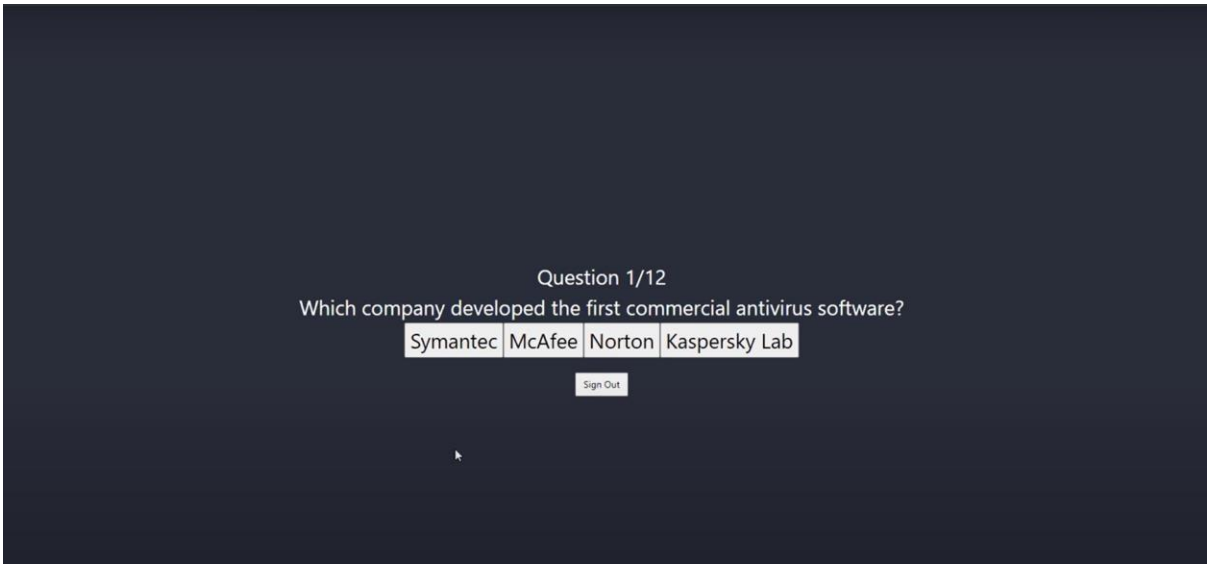
So run:

```
npm install he
npm install -D tailwindcss
npx tailwindcss init
```

Then in “tailwind.config.js”, replace with this code:

```
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Finally, run *"npm start"* and the application will look like this:



Step 5: push local code to GitHub

We have run our frontend locally on localhost using Cognito on the backend. However in the real world it is more likely that you have your code in GitHub or another source control repository, and you have Amplify that pulls from the repo and will host the frontend.

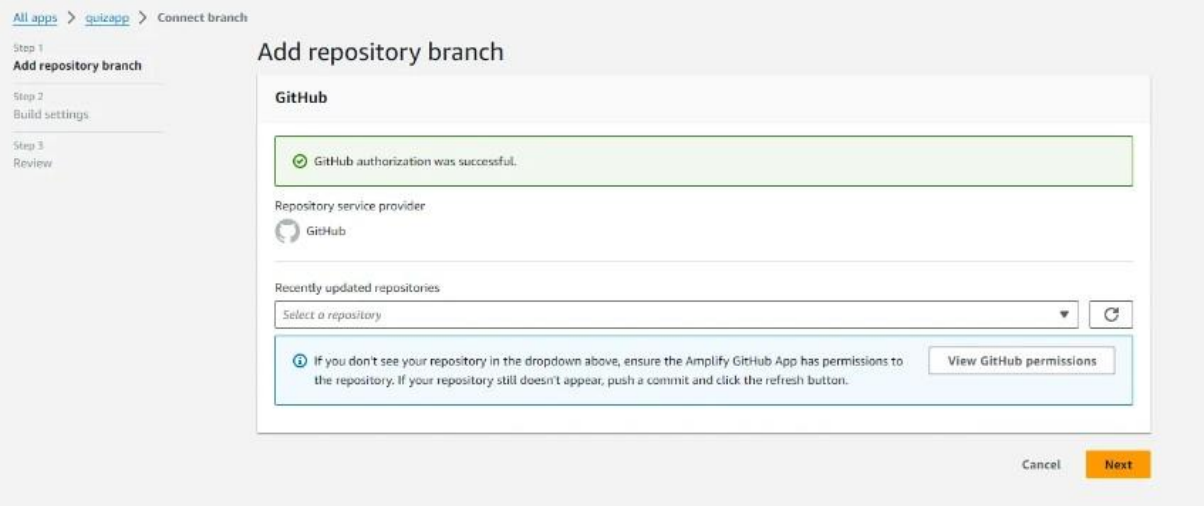
Create a new remote GitHub repository (do not add a README.md file, otherwise you will have to merge) and then run the following commands inside the directory of your React application:

```
git init
git add .
git commit -m "Initial Commit"
git branch -M main
# Here replace with your clone URL
git remote add origin https://github.com/...../amplify-cognito-quizapp.git
git push -u origin main
```

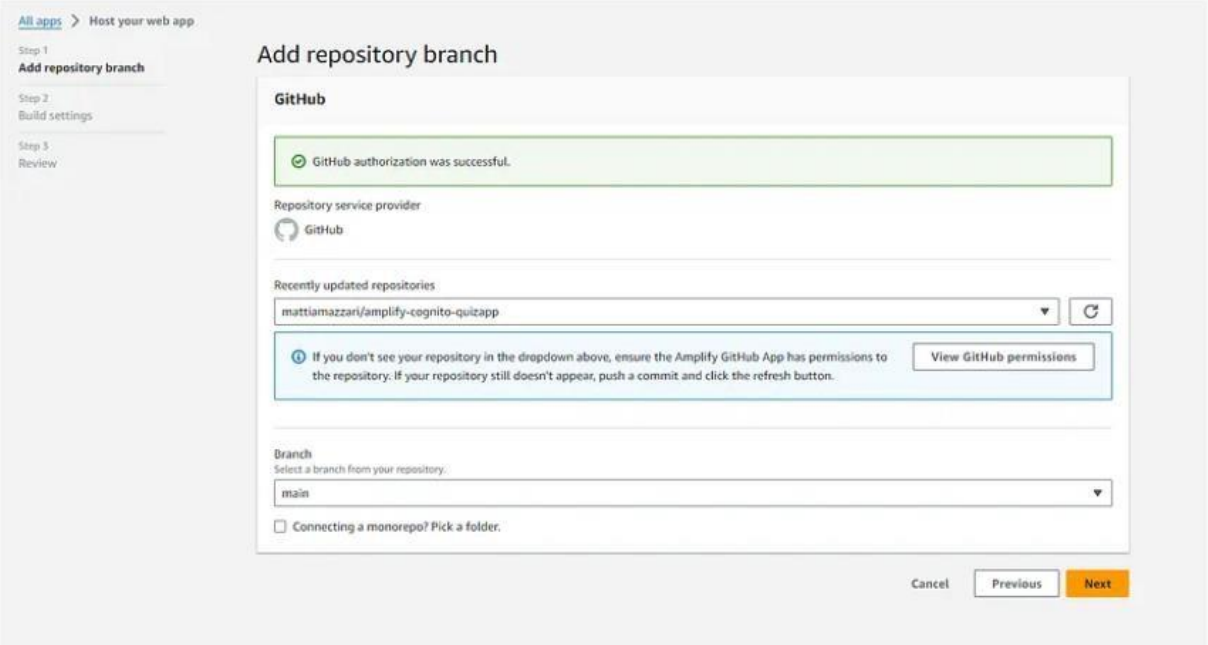
Step 6: set up a CI/CD pipeline

Whenever we make a change to the code in GitHub we want to trigger a new build and deploy in Amplify, so we want to set up a basic implementation of a continuous integration continuous deployment (CI/CD) pipeline.

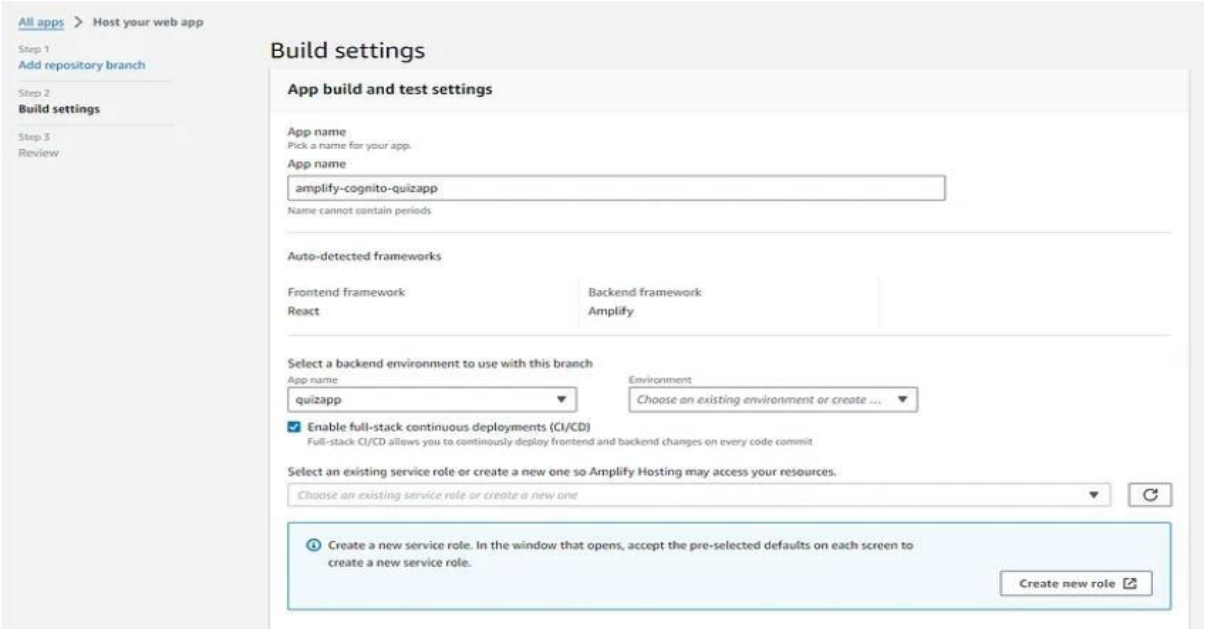
In AWS Amplify in “Hosting environments” is where we say where is our application, select “GitHub” and then “Connect branch”.



If you don’t see your repo, right-click on “View GitHub permissions” and open it in a new tab, a GitHub page will open. Here click on “Only select repositories” and select your repository and click Save.



Then refresh the page in the console and now you should see your repository, then click on Next.



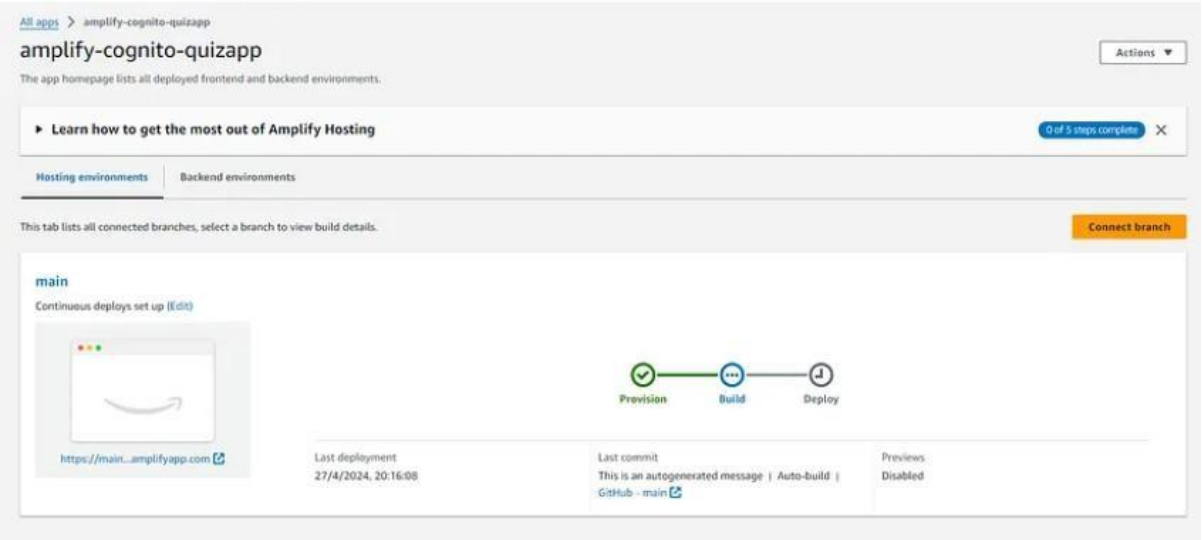
Here you should only change the backend environment, in my case it’s called “quizapp”, it is the name of the app in AWS Amplify, select the envrionment which is “dev”. Also make sure to tick “Enable full-stack continuous deployments (CI/CD).

Then we need to create a service role, so click on “Create new role”. Then click on Next twice and then “Create role”.

Go back to the previous tab and click on the refresh button in the field of the service role.

Then click on Next and then “Save and deploy”.

Now wait few minutes and you should be able to access to your application from the URL similar to “https://main…amplifyapp.com” displayed in the screenshot below under that window image:



Step 7: test the CI/CD deployment

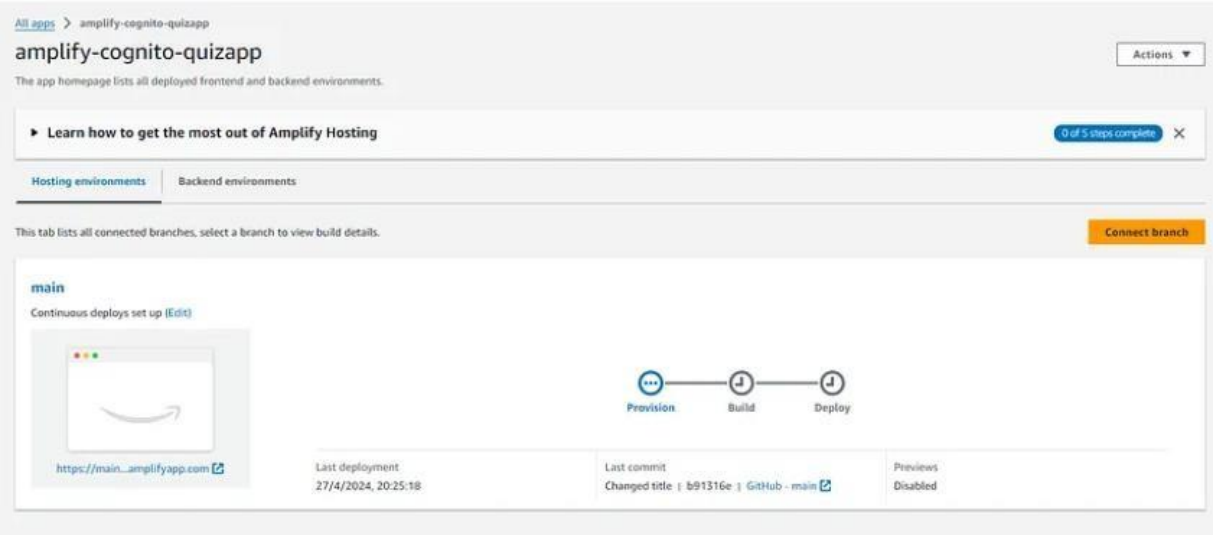
Now if we make a small change to the code, for example changing the title above the questions from “Quiz App” to “Incredible Quiz App” in App.js and push our changes to GitHub, Amplify should detect the change and automatically provision, build and deploy our updated code.

So I apply the change:

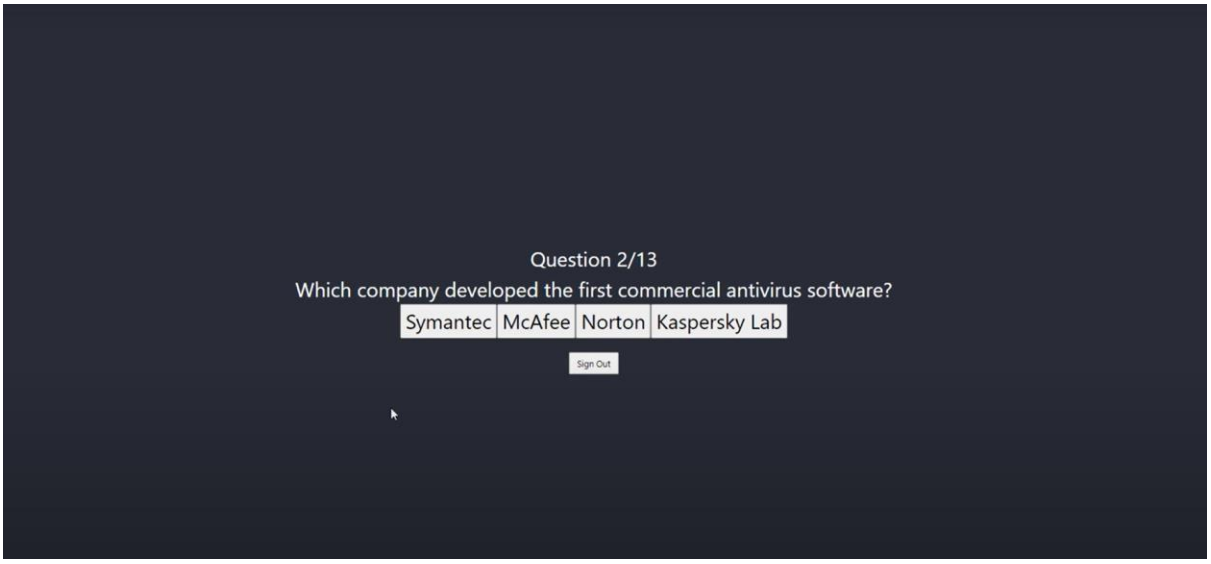
Then run:

```
git add .
git commit -m "Changed title"
git push origin main
```

In the console we will see:



And, after few minutes, we will see the change reflected into our application:



Step 8: clean up

Delete your application in AWS Amplify. Once you've done that, this will automatically remove the associated user pool in Amazon Cognito.

This project is finished, if you liked it please leave a thumbs up and see you in the next project!